***Contents***

* Introduction
* Project Organization
* Development Goals
* Key Features
* Risk Analysis
* Hardware & Software Requirements
* Work Breakdown
* Scheduling
* Workflow
* UML Diagrams
* Monitoring and Reporting Mechanisms
* Appendix
* Summary





This project represents a bold leap into interactive digital entertainment, merging intuitive design with strategic gameplay to deliver an experience unlike any before. It unfolds in two distinct parts—a puzzle and a game—each designed to challenge users to sharpen their reflexes, refine decision-making, and navigate a dynamic environment where every click carries consequence. Through a simple mechanic—fishing—the software draws players into a world of randomized rewards, layered choices, and evolving outcomes.

The journey begins with a familiar digital ritual: opening a webpage, accepting cookies, and logging in. But from there, the experience transforms into a test of attention, timing, and judgment. The first challenge requires players to decipher how to access the game itself. Once inside, they must determine how to initiate gameplay. From there, the player casts their line, chooses when to reel it in, and decides whether to continue fishing or claim their prize. Duplicate popups and randomized gifts introduce complexity, demanding adaptability and precision at every turn.

More than just entertainment, this project is designed to cultivate critical thinking and problem-solving under pressure. It encourages users to weigh risk against reward, persist through uncertainty, and engage with a system built on unpredictability. The developers aim to foster connection, spark curiosity, and leave players both entertained and enriched—with a deeper appreciation for interactive design and the decisions that shape it.





Each of our team members has a designated role tailored to their strengths and responsibilities, allowing for a smooth and efficient Software Development Life Cycle (SDLC). By clearly defining these roles, we ensure accountability, streamline collaboration, and maintain high standards across all phases of development—from planning and implementation to testing and optimization.

| POSITION | NAME(S) | DESCRIPTION |
| --- | --- | --- |
| Project Lead | Emma | The Team Lead (TL) is responsible for assigning documentation and programming tasks to team members based on project requirements. They coordinate across all areas of the program and serve as a liaison between different domains to ensure cohesive collaboration.  The TL also oversees the review and organization of all documentation produced throughout the Software Development Life Cycle (SDLC), ensuring alignment with industry standards and best practices. |
| Lead Frontend | Dakota | The Lead Frontend (LF) developer is responsible for establishing the overall HTML structure of the program with semantic accuracy. They define accessibility standards and ensure consistency across components. The LF also creates and manages CSS style guides and visual hierarchy, overseeing branding, responsiveness, and performance optimization.  In addition, the LF handles error states, loading indicators, and request validations using intuitive and best-practice approaches. They collaborate closely with the Lead Backend (LB) developer to ensure secure and efficient data flow between the frontend and backend. |
| Secondary Frontend | Gabriel | The Secondary Frontend (SF) developer is responsible for building and refining HTML elements such as text boxes, buttons, and message displays. They assist with markup for dynamic content and contribute to layout testing to ensure visual consistency and responsiveness.  The SF applies CSS styling and explores transitions and animations to enhance user experience. They also support The LF in debugging efforts and contribute to the visual polishing of the interface, helping ensure the program meets design and usability standards. |
| Lead Backend | Gregory | The Lead Backend (LB) developer is responsible for setting up and maintaining the foundational frameworks required for the project’s product. This may include, but is not limited to, technologies such as Flask, Docker, and Django.  They design backend logic to handle user input, manage data flow, and route responses efficiently. The LB also establishes coding and security standards, and makes key architectural decisions to ensure a stable, scalable, and high-performing backend infrastructure. |
| Secondary Backend | Dakota | The Secondary Backend (SB) developer supports backend optimization and performance tuning, ensuring smooth and efficient operation of the system. They handle tasks assigned by the Lead Backend (LB), which may include monitoring bot responses, updating backend functions, and refining logic based on user interactions.  The SB also manages integrations with external services or APIs as instructed, ensuring secure and reliable data exchange. |
| Database Admin | Emma | The Database Admin (DBA) is responsible for designing and managing the overall database architecture. This includes creating and maintaining schemas, tables, relationships, constraints, and keys to support the program’s functionality and scalability.  They ensure data integrity, normalization, and cleanliness across all environments, while coordinating with the frontend team to support accurate and efficient fetch() calls and data rendering. The DA also assists the backend team by writing and optimizing queries, joins, and context-specific data logic. |
| Secondary Database | Gabriel | The Secondary Database Admin (SDA) assists in implementing and refining the database schema, supporting the design of tables, relationships, and constraints. They contribute to data validation and cleanup efforts to ensure accuracy, consistency, and usability across the system.  The SDA helps troubleshoot frontend/backend data flow issues, working closely with both teams to resolve mismatches and latency concerns. They document endpoints, query formats, and expected responses to support clear communication and efficient development.  Throughout the development process, the SDA tests queries, flags risks or inconsistencies in the data structure, and contributes to performance optimization. |
| Testers | All Team Members | All team members are responsible for thoroughly testing their individual domains to ensure functionality, usability, and adherence to project standards. In addition, each member participates in integration testing to verify smooth interaction between components across the frontend, backend, and database layers. |



For a project condensed into such a short timeframe, we have identified a set of clear development goals. While it may not be possible to implement or polish every feature we have in mind, our aim is to establish and achieve certain key benchmarks that define the core experience of the game. These goals will serve both as milestones for progress and as measures of success for the final product.

| * Create an effectively obnoxious but fun pre-game that challenges players to think outside the box about how they access and progress through the game itself. |
| --- |
| * Create the core gameplay loop by implementing a fishing mechanic where players cast a line and earn a random reward. |
| * A loot-style system that ensures each fishing attempt produces unpredictable outcomes. |
| * Optimize UX clarity to ensure goals are obvious even if not explicitly stated |
| * Prioritize core functionality to ensure a complete product may exist. |
| * Deliver a playable prototype within the designated timeframe. |





In addition to the primary goals necessary for a complete and sufficiently playable version of the project, there are several additional key features we aim to implement. These features will enhance the overall experience, adding depth, variety, and polish to the program beyond its core functionality.

| * An unconventional entry sequence where players must navigate fake logins, cookie prompts, and misleading popups before reaching the game, previewing the thought-provoking nature of the game itself. |
| --- |
| * Duplicate windows and interruptions that challenge focus and adaptability, such as fake advertisements, random images, or other random events. |
| * Pop-ups that distract the player selected from a bucket. |
| * An economy and store with power-ups that can quicken how fast you catch fish. |



Our team has taken a proactive approach to risk management by identifying potential challenges across three broad categories: technological, environmental, and personal. Technological risks include issues related to frameworks, tools, and system performance; environmental risks account for natural events or disruptions that could impact workflow; and personal risks reflect individual availability and unforeseen circumstances. By recognizing these areas early, we’ve positioned ourselves to respond effectively and maintain momentum throughout the development process. Specific risks and mitigation strategies are outlined in the table below.

| RISK | DESCRIPTION | RISK PRIORITY/RISK EFFECT | RISK PLANNING |
| --- | --- | --- | --- |
| Late Delivery of Project | The project is not submitted by the deadline or is incomplete at the time of submission. | Low / Catastrophic | To prevent late delivery of the project, our team will follow internal deadlines, conduct regular progress check-ins, and maintain clear documentation to ensure continuity and timely completion. |
| Cloud Software Failure | Online platforms used for collaboration (e.g., Google Docs, GitHub) experience outages or data corruption. | Low / Catastrophic | To mitigate cloud software failures, our team will maintain local backups of critical files, establish alternative communication channels, and schedule regular syncs to reduce dependency on any single platform. |
| Team Members Unavailable | One or more team members are unavailable during critical phases (e.g., meetings, coding tasks) due to unforeseen events such as illness or emergencies. | Low / Serious | To address potential team member unavailability, we’ve implemented role redundancy, shared documentation practices, and flexible scheduling to ensure that progress can continue smoothly even if someone is unexpectedly absent. |
| Hardware Failure | A team member’s device (e.g. PC, laptop, monitor) malfunctions, disrupting their ability to contribute to the project. | Moderate / Serious | To mitigate hardware failure, our team will ensure critical files are backed up to the cloud, maintain access to alternative devices when possible, and communicate promptly to reassign tasks or adjust timelines if disruptions occur. |
| Project Concept Modifications / Scope Creep | The client or team members alter the project concept beyond the original scope, making it unmanageable within the given timeline and resources. | Moderate / Serious | To manage project concept modifications and scope creep, our team will maintain a clearly defined scope document, require consensus for major changes, and evaluate all proposed adjustments against timeline and resource constraints before implementation. |
| Local Software Failure | Software installed on personal devices (e.g. VS Code, Discord, etc.) becomes unusable, affecting the communication or development of the project. | Moderate / Tolerable | To mitigate local software failures, our team will ensure access to alternative tools or platforms, maintain cloud-based backups of work, and communicate promptly to adjust workflows or reassign tasks as needed. |
| Specified Tasks Not Complete by Deadline | Assigned tasks are not completed on time, delaying subsequent phases of the project. | Low / Tolerable | To reduce the risk of tasks not being completed by their deadlines, our team will set clear expectations, use progress tracking tools, and maintain open communication to identify delays early and adjust workloads or timelines as needed. |
| Debugging Takes Longer than Anticipated | Errors encountered during debugging take longer than expected, impacting the overall project timeline. | Moderate / Serious | To mitigate delays caused by extended debugging, our team will allocate buffer time in the project schedule, prioritize early testing, and collaborate closely to resolve issues efficiently and prevent bottlenecks. |
| Natural Disaster | A natural disaster (e.g. tornado, hurricane, blizzard) disrupts access to local or cloud-based tools, halting progress for an extended period of time. | Low / Catastrophic | To prepare for natural disasters, our team will prioritize cloud-based backups, maintain flexible timelines, and ensure remote access options are available to minimize disruption and support continuity during extended outages. |
| Team Member Reisigning from Project | A team member voluntarily leaves the project before its completion, affecting workload distribution, available skills, and overall progress. | Low / Serious | To prepare for the possibility of a team member resigning from the project, we’ve implemented shared documentation, cross-training on key tasks, and flexible role coverage to ensure that progress can continue smoothly and responsibilities can be redistributed effectively. |
| Poor Management | Ineffective task allocation or unrealistic timelines hinder team productivity and project success. | Low / Serious | To mitigate poor management risks, our team will prioritize clear communication, realistic scheduling, and collaborative task allocation to ensure that responsibilities are well-distributed and timelines remain achievable throughout the project. |
| Security Breach/Data Leak | Sensitive project data is exposed due to poor security practices or external attacks. | Low / Catastrophic | To mitigate the risk of a security breach or data leak, our team will follow secure coding practices, use trusted platforms with encryption, and limit access to sensitive information through role-based permissions and regular audits. |
| Version Control Conflicts | Merge conflicts, improper use of Git, or insufficient documentation of changes results in lost work or delays. | Moderate / Tolerable | To prevent version control conflicts, our team will follow consistent Git workflows, document changes clearly in commit messages, and conduct regular syncs and code reviews to catch issues early and maintain repository integrity. |
| Client Communication Breakdown | Misalignment or lack of feedback from the client causes confusion or rework. | Moderate / Serious | To prevent client communication breakdowns, our team will maintain consistent contact through scheduled updates, clarify expectations early, and document all feedback to ensure alignment and minimize the need for rework. |

With a clear understanding of potential obstacles and a set of targeted mitigation strategies in place, our team is equipped to navigate challenges confidently and efficiently. This structured planning not only strengthens our ability to stay on track but also reinforces collaboration, adaptability, and accountability throughout the project. By anticipating disruptions and preparing thoughtful responses, we’ve created a realistic and sustainable path toward successful completion.



To support both users and developers, systems must run a currently supported OS (MacOS or Windows) with stable internet access. End users need only a modern browser and basic hardware (dual-core CPU, 4 GB RAM), while developers require more robust setups (8 GB RAM minimum) and tools like VS Code, Git, and Docker. Development environments should include Django with a virtual environment, API testing tools, and a database engine. Client-side work relies on HTML/CSS/JavaScript and GitHub for collaboration.

## Hardware Requirements

**Both:**

- A computer with an operating system that is still supported by their respective computer manufacturer (MacOS by Apple, Windows by Microsoft) for consistent performance & security enhancements as well as bug fixes. Internet access will also need to be readily available.

**End users:**

- Minimum hardware should be: Dual-core CPU, 4 GB RAM, 1 GB free disk space. For an even better experience, it is recommended to have a quad-core CPU, 8 GB RAM, 2 GB+ free disk space

**Development Team:**

- Minimum hardware should be: Computers with at least 8 GB RAM, dual-core or better CPU.

- Aforementioned internet access needs to be stable for consistent connections to the workspace as well as GitHub repo synchronizations.

Software Requirements

**End Users:**

- No additional outside of the modern web browser is needed for accessibility.

**Development Team:**

**Server-side:**

- Virtual environment (such as venv) to host Django Framework, and other dependencies Python requires.

- Thunder Client plugin for API testing.

- SQL or SQLite database engine.

- Docker (or similar) for deployment

**Client-side:**

- Latest version of VS Code (this is the chosen IDE) and all required plugins.

- Git for version control.

- An accessible GitHub repository for collaboration.

- All necessary HTML/CSS/JavaScript files for proper UI/UX.





Roadmap/Work Breakdown

The roadmap directly reflects the work breakdown by organizing individual tasks into a structured, phase-based layout. Each major component of the work breakdown, planning, setup, development, integration, and finalization was translated into a roadmap column, with tasks broken down into actionable items. For example, the work breakdown identifies activities such as building the HTML skeleton, establishing the backend framework, and defining the database schema. In the roadmap, these appear under the setup phase, giving the team a clear visual representation of when and how these tasks should occur in the project timeline.

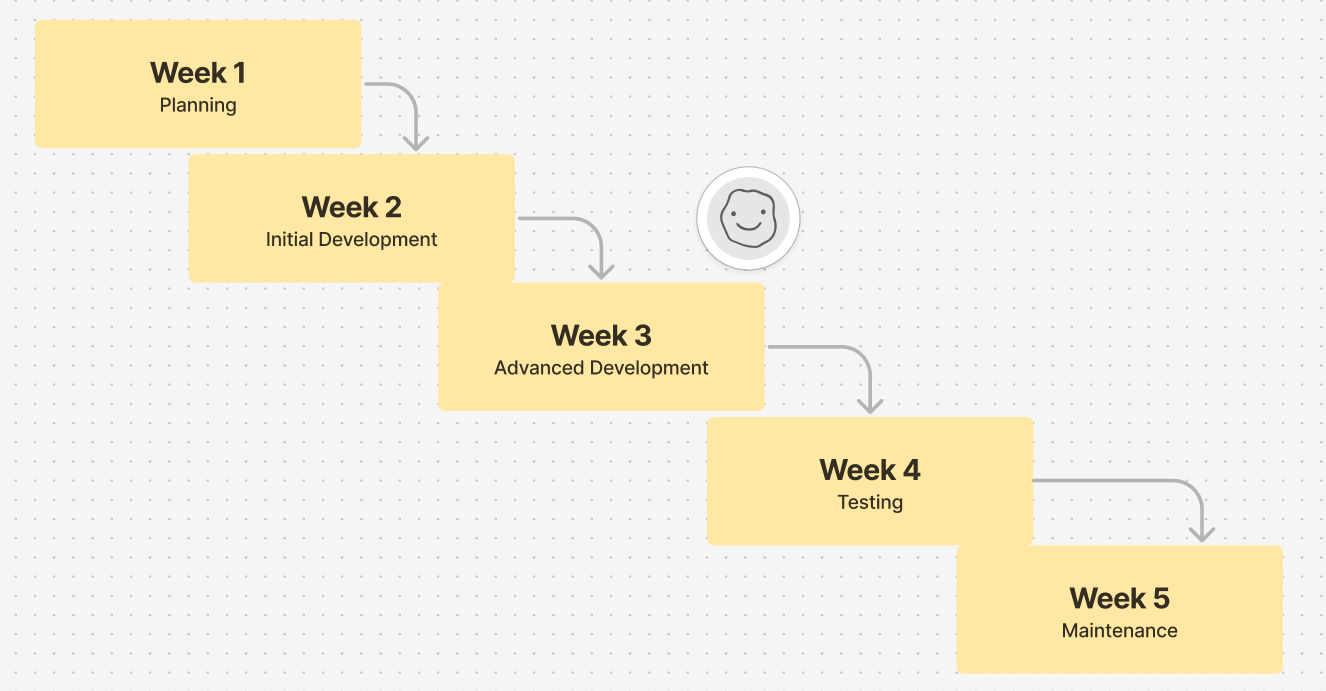
By aligning the roadmap with the work breakdown, the project gains both clarity and accountability. The work breakdown ensures no major area of development is overlooked, while the roadmap contextualizes these tasks within a logical progression. This makes it easier to track milestones, manage dependencies, and confirm that responsibilities are evenly distributed across team members. Together, the roadmap and work breakdown create a cohesive plan that balances detail with direction, guiding the team from initial planning all the way through testing, polish, and delivery.

See the referenced roadmap on pg.15.

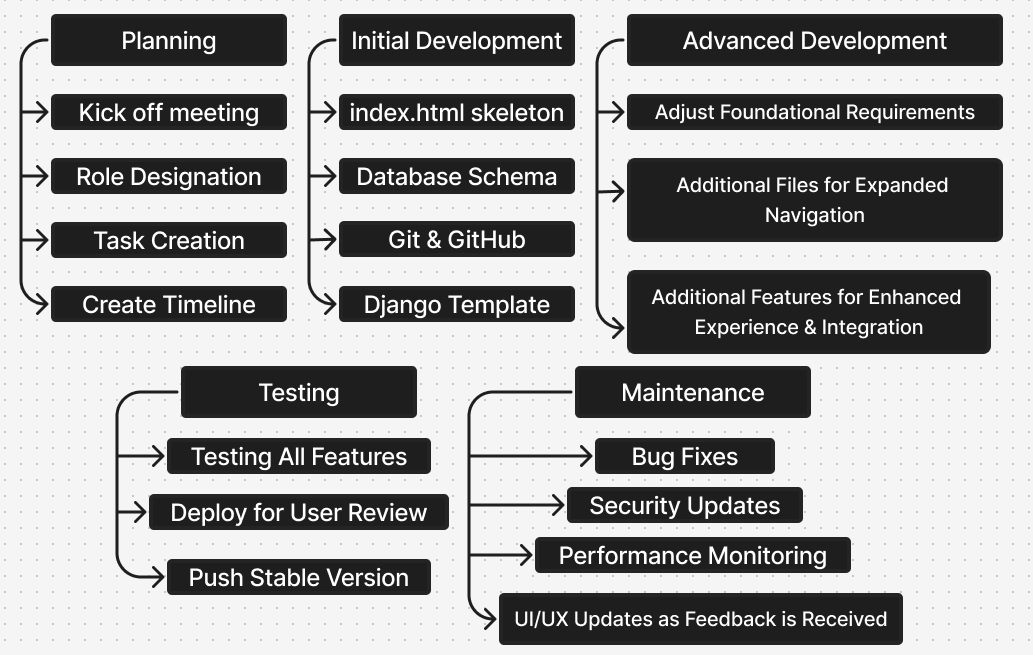
| **Planning**  **(Phase 1)** | **Setup**  **(Phase 2)** | **Development**  **(Phase 3)** | **Integration**  **(Phase 4)** | **Finalization**  **(Phase 5)** |
| --- | --- | --- | --- | --- |
| **Kickoff Meeting & Role Assignments**  *Members:* All Team Members  **Setup Shared Workspace (VS Code + GitHub)**  *Members:* Backend + Frontend  **Define Project Scope & Deliverables**  *Members:* Project Lead  **Create Documentation Template (Google Docs/Markdown)**  *Members:* Project Lead + DBA | **Build Initial HTML Skeleton (index.html)**  *Members:* Frontend  **Define Database Schema & Relationships**  *Members:* DBA + Backend  **Establish Backend Framework (Django/Flask)**  *Members:* Backend  **Setup Version Control Workflow (Branching, Pull Requests)**  *Members:* Backend + Frontend  **Setup Local Development Environments (Virtual Env, Dependencies)**  *Members:* All Team Members | **Implement CSS Style Guide & Accessibility Standards**  *Members:* Frontend  **Develop Frontend Components (Buttons, Forms, Layouts)**  *Members:* Frontend  **Create Authentication Flow (Login, Fake Cookies/Popups)**  *Members:* Backend + Frontend  **Develop Database Queries (CRUD, Fetch)**  *Members:* DBA + Backend  **Write Unit Tests for Core Features**  *Members:* All Team Members | **Connect Frontend to Backend (fetch calls)**  *Members:* Backend + Frontend  **Integration Testing (Frontend/Backend/Database)**  *Members:* All Team Members  **Bug Tracking & Fixes (Issue Log)**  *Members:* All Team Members  **Cross-Browser Testing (Chrome, Firefox, Edge)**  *Members:* Frontend | **Polish UI/UX, Add Animations & Visual Consistency**  *Members:* Frontend  **Final Testing & QA Review**  *Members:* All Team Members  **Documentation Review (Technical + User Guide)**  *Members:* Project Lead + DBA  **Prepare Project Presentation/Demo**  *Members:* All Team Members  **Submit Final Project**  *Members:* Project Lead |



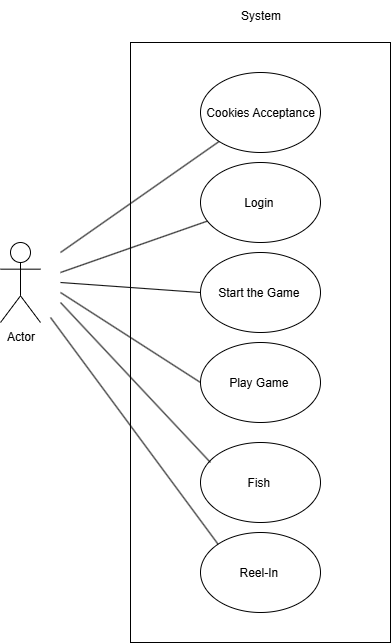
The project will be completed on a five week timeline. Week 1 held our kick-off meeting and was our planning phase. Week 2 will begin our initial development, creating skeleton templates and getting the foundational integrations in check. Week 3 will be heaviest as this will be the advanced development phase where we will collaboratively come together to synchronize all areas of the project together. Week 4 will be testing our recently created project, hunting for bugs and looking for feedback from users. Week 5 will be fixing the bugs that are found during testing, making adjustments as necessary per user feedback as well as any security updates that are needed.











**Use-Case Diagram**

This use case diagram outlines the foundational user interactions for our gag-based game application, designed to foster engagement and intuitive problem solving through interactive play. The roadmap begins with **Cookies Acceptance** and **Login**, each functioning as standalone puzzles that introduce the player to the game’s quirky logic. Once ‘authenticated,’ users begin the **Start the Game** process, triggering a reaction-based, mechanically driven puzzle that sets the stage for deeper interaction. From there, the core loop—**Play Game**, **Fish**, and **Reel-In**—guides the player through progressive stages of gameplay, each building on the last to reinforce curiosity, timing, and playful experimentation.



This project will introduce a new breed of innovation that no previous generation has ever accomplished. The project will introduce to the users how to enhance their ability to think quickly in dire situations as well as the ability to decide what choices are best for the current situation.

**Activity # Description Estimated Time to Complete**

1 Kickoff meeting & role assignments 2 hours

2 Set up shared VS Code workspace & GitHub repo 3 hours

3 Build initial HTML skeleton (index.html) 4 hours

4 Define database schema & relationships 5 hours

5 Establish backend framework (Django/Flask setup) 6 hours

6 Implement CSS style guide & accessibility standards 5 hours

7 Develop frontend components (buttons, forms, layouts) 6 hours

8 Connect frontend to backend via fetch() calls 6 hours

9 Integration testing across frontend/backend/database 4 hours

10 Polish UI/UX, add animations & visual consistency 5 hours

11 Final testing, bug fixes, and documentation review 6 hours.



Overall, the main activities planned for the project alongside their estimated duration can be found here.

| **Activity** | **Short Description** | **Time Estimate** | **Dependencies** |
| --- | --- | --- | --- |
| 1 | Project design document | 1-2 days | - |
| 2 | Design and set-up framework for modular, reusable functions (Puzzle templates, pop-up defaults, loot-box table) | 1 day | 1 |
| 3 | Develop fishing clicker core loop (casting, reeling, reward tracking) | 1 day | 2 |
| 4 | Create bucket of different puzzles, pop-ups, and related events using framework | 4-6 hours | 2,3 |
| 5 | Integrate risk/reward choices | 4 hours | 2,3 |
| 6 | Properly optimize and design user-friendly UI | 6 hours | 4,5 |
| 7 | Playtesting & iteration | 2-3 days | 6 |
| 8 | Documentation | 2 hours (per doc) | - |



Our team’s goal is to build a gag website—a random web game designed to both entertain and lightly annoy users in fun, unexpected ways. Rather than focusing on a traditional client project, this one is all about creativity, humor, and experimenting with unique interactions on the web.

Each team member has a clearly defined role to keep development organized: Emma is our Project Lead and Database Admin, coordinating the overall effort and keeping documentation on track. Dakota is taking point as Lead Frontend, setting up structure, styling, and usability standards, while also supporting Gregory as Secondary Backend. Gregory is our Lead Backend, handling frameworks and system logic, while Gabriel supports Dakota on the frontend and Emma on database work. Testing responsibilities are shared across all members to make sure the game works smoothly and consistently.

We’ll achieve our goals by following a structured development process: starting with planning and role assignments, then moving into frontend structure and styling, backend logic and frameworks, and database setup. As the project evolves, each role overlaps where collaboration is needed—for example, frontend and backend integration, or database queries supporting UI interactions. Regular testing and documentation will keep the project stable, while our collective focus remains on delivering a quirky, entertaining experience for users.